



FIG.1

REAL-TIME GATEKEEPER

```

REPEAT:
--If the cycle-time has expired, /* then getting ready for the next cycle by re-setting variables */
then do; C = C + 1;
    Get Nunused (j) from WWR Assistant;
    set Cadmitted(i,j) = Cadmitted(i,j) + Nadmitted(i,j) for all (i,j);
    set Crejected(i,j) = Crejected(i,j) + Nrejected(i,j) for all (i,j);
    set Nadmitted(i,j) = 0 for all (i,j); set Nrejected(i,j) = 0 for all (i,j);
    set Nsharable(j) = Nexcess(j) + Nunused(j) for all (j);
end;

(STEP 2) --Get a packet;
    If the packet's SYN bit is OFF
    then do: let the packet pass through; go to REPEAT; end;
    /* handling TCP connection request */
    Find its index (i,j);
    --If Nadmitted(i,j) < Nmin(i,j) /* guaranteeing the minimum SLA */
    then do: set Nadmitted(i,j) = Nadmitted(i,j) + 1;
        let the packet pass through;
        go to REPEAT;
    end;

(STEP 4) --If (Nadmitted(i,j) ≥ Nmin(i,j) & (Nadmitted(i,j) < Nmax(i,j) & (Nsharable(j) > 0)
    /* the request can be admitted */
    then do; set Nadmitted(i,j) = Nadmitted(i,j) + 1; Set Nsharable(j) = Nsharable(j) - 1;
        let the packet pass through;
        go to REPEAT;
    end;
    /* the request should be rejected */
    else do; set Nrejected(i,j) = Nrejected(i,j) + 1;
        forward the packet to WWR Guide;
        go to REPEAT;
    end,

```

FIG. 2

ASSISTANT FOR REAL-TIME GATEKEEPER

```

set Numused(j) = 0 for all (j);
If the mode of operation is "not borrowing" then return;
/* now the mode of operation is "borrowing", and thus need to compute Numused(j) for all j */
For every (j) and for every (i),
    If Nadmitted(i,j) < Nmin(i,j) then set Numused(j) = Numused(j) + (Nmin(i,j) - Nadmitted(i,j));
For every (j), Numused(j) = Numused(j) * Unused_Permit_Factor(j);
return;

```

FIG. 3

TARGET-RATE BASED GATEKEEPER

```

/* Rate based Gatekeeper algorithm */
REPEAT:
--- If the cycle-time has expired, /* then getting ready for the next cycle by re-setting variables */
    then do; C = C + 1;
        set Cadmitted(i,j) = Cadmitted(i,j) + Nadmitted(i,j) for all (i,j);
        set Crejected(i,j) = Crejected(i,j) + Nrejected(i,j) for all (i,j);
        set Nadmitted(i,j) = 0 for all (i,j); set Nrejected(i,j) = 0 for all (i,j);
    end;
---Get a packet;
    If the packet's SYN bit is OFF
    then do: let the packet pass through; go to REPEAT; end;
/* handling TCP connection request */
---Find its index (i,j);
    If Nadmitted(i,j) < Ntarget(i,j)
    /* accepting the request */
    then do; set Nadmitted(i,j) = Nadmitted(i,j) + 1;
        let the packet pass through;
        go to REPEAT;
    end;
/* rejecting the request */
    else do; set Nrejected(i,j) = Nrejected(i,j) + 1;
        forward the packet to WWR Guide;
        go to REPEAT;
    end;

```

FIG. 4

ASSISTANT FOR TARGET-RATE-BASED GATEKEEPER

```

/* compute Ntarget(i,j) */
(STEP 1) --- Do the following for every application class j;
              If ((Nrejected(i,j) = 0)
                & (Nadmitted(i,j) <= Nmax(i,j)) over all i
                then go to STOP;
              COMPUTE_TARGETS:
(STEP 2) --- Let Ntarget(i,j) = Nadmitted(i,j) + Nrejected(i,j) for all i;
              Compute Ntotal(j); /* Ntotal(j) here is the sum of Ntarget(i,j) over all i */
              For every (i,j) such that Ntarget(i,j) > Nmax(i,j) /* demand is exceeding the maximum */
                first set Ntotal(j) = Ntotal(j) - (Ntarget(i,j) - Nmax(i,j)); /* adjusting the total demand */
                And then set Ntarget(i,j) = Nmax(i,j); /* not allowing the demand to exceed the maximum */
              If Ntotal(j) <= Nlimit(i,j) then go to STOP;
              /* each Ntarget(i,j) needs to be set below Nmax(i,j) */
(STEP 3) --- Let Nexceeded(j) be the sum of (Ntarget(i,j) - Nmin(i,j)) over those i such that Ntarget(i,j) > Nmin(i,j);
              If the-mode-of-operation is "non borrowing" then set Nsharable(j) = Nexcess(j);
              If the-mode-of-operation is "borrowing" then compute Numused(j) and set Nsharable(j) = Nexcess(j) +
                Numused(j) * Unused_Permit_Factor(j);
              /* What performed next is fair allocation. However, priority based allocation can be performed instead */
              For every i such that Ntarget(i,j) > Nmin(i,j)
                Set Ntarget(i,j) = Nmin(i,j) + (Ntarget(i,j) - Nmin(i,j)) * (Nsharable(j) / Nexceeded(j));
                else set Ntarget(i,j) = Nmax(i,j);
(STEP 4) --- STOP:
              End;

```

FIG. 5